Building Clinical Concept Embeddings with computing resource constraints

Proposed by
A3.AI
Xue.Yang@a3.ai
Antonio.Linari@a3.ai
Changrong.Ji@a3.ai

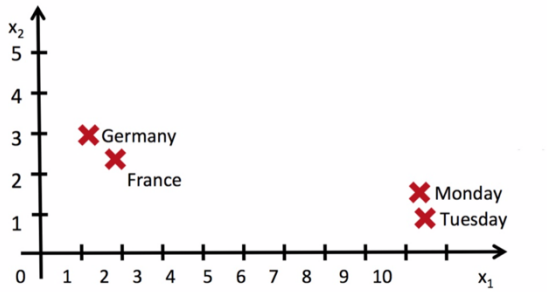
Research Aim

Medical claim data is notoriously sparse and noisy with high dimensionality. There are over 100,000 distinct diagnosis, procedure and drug codes. As categorical features these are too fine-grained for model training. Our goal is to build embeddings from billions of claims data points by projecting clinical concepts into 50-100-dimension vector spaces. This not only reduces the features' dimension but also better captures the nuanced “relatedness”, resulting in more predictive features with computing resource constraints

Study Design

1. Clinical concept embedding

By its very nature, health care data is complex. In this study, we used medical claims data, it provides us a good amount of patient history, at the meantime, the high dimension of all the medical billing codes makes the dataset bloated and clumsy. In order to overcome these disadvantages and uncover the value of medical claims data, we are borrowing word embedding technique to our system. Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP). A great way to represent sparse, high-dimensional words or phrases in the vocabulary to a lower dimension using vectors of real numbers. In this low-dimensional space, words of similar meaning are located near each other in the embedded vector, and the relative location of two words in the space could encode a meaningful relationship. In this picture,



we could see, Germany and France are both countries, they located near each other, and so dose Monday and Tuesday (Fig.1).

Fig.1 low dimensional representation of words. (edited from Bengio, "Representation Learning and Deep Learning", July. 2012. UCLA

Compare with those words in language, medical bill codes are also high-dimensional data. There are over 100,000 distinct diagnosis, procedure and drug codes. We are proposing a clinical concept embedding here to represent the code to a low-dimensional space using vectors. We are using unsupervised learning to capture the relationship between different diagnosis codes, procedure codes and drug codes, the embedding model, could not only use for risk prediction for specific task, but also could use for transfer learning on other tasks.

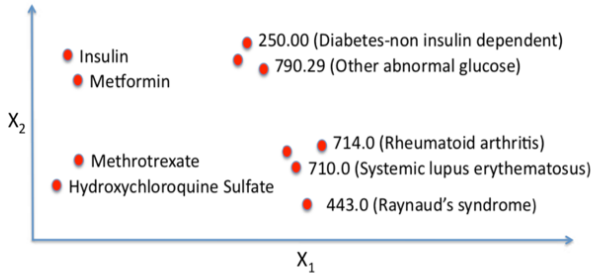


Fig.2 Illustration a low-dimensional representation (in this case, 2-dimensions) of medical concepts. Similar concepts are close to each other in Euclidean space.

A medical concept embedding model could do the similar thing. Chio and his colleges [1] trained an embedding model and plot these codes or clinical concept in to a 2-dimensional space (Fig.2), we could see, the codes have close relationship are located closely. In this way, we can efficiently reduce the dimension of our data and extract features without supervise.

2. Graph Embeddings applied to Healthcare

Embeddings are an attempt to reduce a high-dimensional vector space into a relatively low-dimensional space. A typical use of embeddings is in Natural Language processing where we want to represent words as features vectors for classification, prediction or extraction. The simplest way to transform words into vectors is by creating a sorted dictionary D={w0,w1,w2….} and then represent each word wi as a vector of all zeros but the column i matching the position of w in D where we put the value 1 (e.g. w3={0,0,0,1,0….}). This type of representation is called one-hot encoding and, even if it's easy to create it has some drawbacks:

a. it's easy to imagine that for a very large dictionary of size N >> 0 we end up with a N vectors each of dimension Nx1 and very sparse (all zero but one);
b. every word w_i is orthogonal to w_j (with i <> j), meaning that all the relations existing between words are lost.

A better way to represent words is by using the context those words are surrounded. And this is what embeddings are supposed to do in order to represent meaningful vectors.
Usually the context is defined by capturing the proximity of words. So, given a sequence of words w_0,w_1,w_2,w_3…, the idea is to predict, for example, the word w_i given w_{i-1} and w_{i+1} and do so for all the words in our dataset (e.g. word2vec).
In order to be able to predict any word given a sequence of other words, it's necessary to train a so-called *auto-encoder*.

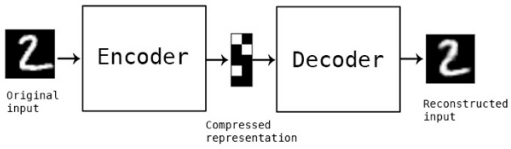


Fig.3 Auto-encoder

An auto-encoder is a neural network that given an input I, tries to compress it as much as possible so to be able to reconstruct it on the output, minimizing the loss. In the above picture we want to find a compressed representation of the input pictures so that a decode is able to reconstruct it as exactly as possible.
In case of words the input can be a sequence of words that we want to reproduce exactly in the output or a sequence of words to predict another word.
The sequence of input words is still represented by one-hot encoding vectors and so the output, but between the encoder and the decoder we now have a compressed representation of that sequence that we can use as an embedding
But a text is not only a sequence of words. Every word has a specific meaning and some words are more relevant than others, some words represent entities like people names, organizations or places. Words have proximity relationship like subject/action/object but they can have also "long distance" relationship for example between two close sentences. For this reason, it's possible to find better representation by using more complex data

structure. A data structure that is very successful is the graph. Representing words as nodes and relationships as edges of a network graph:

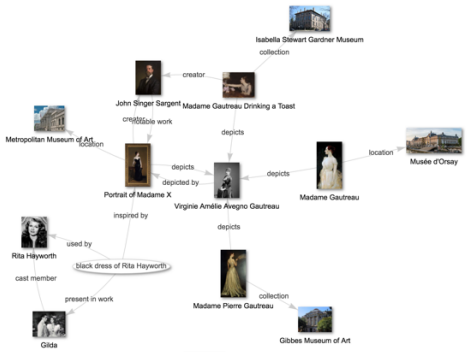


Fig.4 Knowledge Graph - Wikimedia Commons Courtesy

If we wanted to represent each node as a vector, now we can take into account direct and indirect relationships present in the graph. A very simple but quite effective technique is to use random walks [2]. A random walk in a graph is a path connecting one node to another through the selection of random adjacent path:

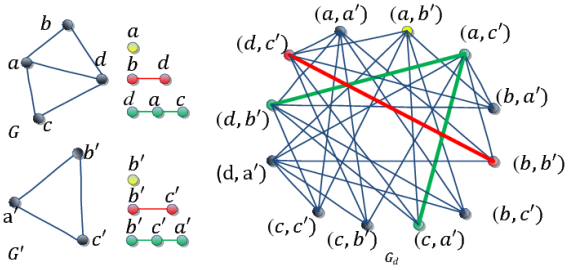


Fig. 5 Random Walks

In the case of text, every walk is a sequence of words that can be use with the already described auto-encoder to create a more expressive representation that captures also extra-proximity relationships.

As this said, a novel approach to represent diseases, prognosis and drugs as embeddings is based on the creation of a medical knowledge graph. Such knowledge graph is obtained from claims and other data (e.g. drugs and social data). The embeddings calculated using a medical knowledge graph, can take into account claim data, social data, drag data and any other related data available. This medical knowledge graph is created by transforming current table related to patients/claim into a graph, according to the following available fields:

- Payer name
- Patient token
- Relation (single, married, …)
- Gender
- Race
- Year of Birth
- Zip Code (3 digits)
- State
- Primary Diagnosis
- Other diagnosis
- Conditions (if available)
- Covid-19 positive

In the case of COVID-19 patients also other data are available:

- ECMO
- ICU
- Ventilator

The above data represent classes of nodes in a graph where the Patient is the node all the other classes relate to. Basically, every single claim is represented by a STAR where the Patient Token is in the center (colors and radius can represent diverse dimensions):

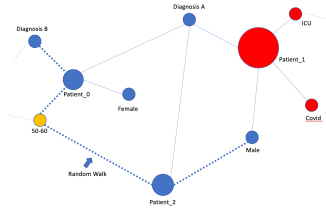


fig.4 a simplified visualization of the knowledge graph

The knowledge graph is big enough to represent random walks as kind of sentences in a text where nodes are linked together through a direct relationship.
Once collected many random paths it's possible to run algorithms like word2vec or glove (in case of limited resources) or more powerful algorithms like Transformers.
The choice to manage every class or attribute as Node helps to control the random walks so to skip specific classes if needed (like gender or race) to avoid for example bias or for privacy reasons.
The beauty of this approach is that is not necessarily CPU/memory intense and fits very well in the available limited resources environment. Respect to more common approaches, with a knowledge graph it's possible to capture multiple aspects and shape every node embedding taking into account multiple dimensions.

3. The process

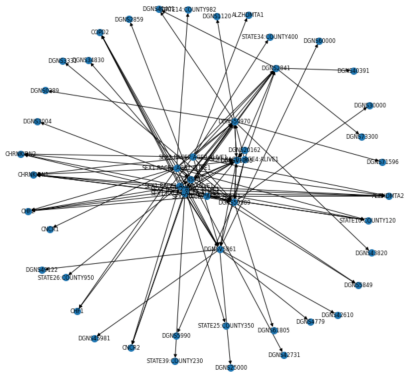
For creating the described embeddings we used synthetic data coming from CMS and relative to the time period 2008-2010. Diagnosis and prescriptions codes are still compliant with ICD9 taxonomy.
In particular we used inpatients claims and beneficiary info:
From beneficiary:

- Gender
- Race
- Year of Birth
- Year of Death
- State and partial ZIP code
- Preconditions

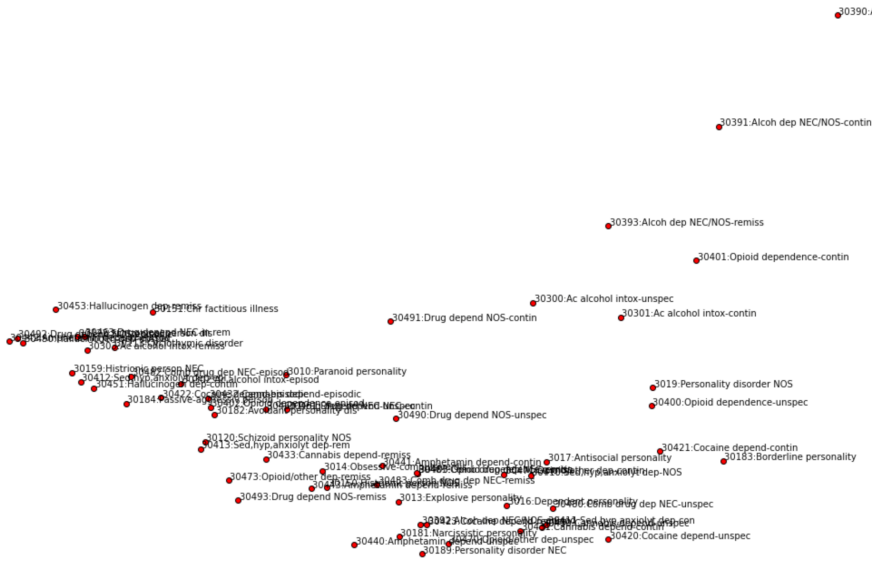
From claims:

- Diagnosis
- Prescriptions
- Drugs (if any)

Every patient has been identified by a fingerprint patient_fingerprint = Gender:Race:Age:Alive where age refers to the age range given the YOB (0-10 = 0, 10-20 = 1 etc) and Alive is 0 or 1 depending if a person was alive or not at that time. Every location has been identified by a fingerprint location_fingerprint = State:ZIP (3 digits).
The knowledge graph has been built connecting every patient fingerprint with location_fingerprint, all preconditions and all diagnosis, prescriptions and drugs.



Node2Vec was applied to the graph and how it's possible to see, the generated embeddings for DIAGNOSIS are correctly clustered:



Data

De-identified personal level data provided pro bono by COVID-19 Research Database Consortium
Base Population includes:
- 90 million patients 7 years of medical claims history, over 3 billion claim lines
- 40 million patient's outpatient EHR records
- 240 million people's Social Data
- Death Registry

Reference:

1. Choi, Y., Chiu, C., and Sontag, D., Learning Low-Dimensional Representations of Medical Concepts AMIA Jt Summits Transl Sci Proc. 2016; 2016: 41–50.
2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5108654/
3. https://www.cms.gov/Research-Statistics-Data-and-Systems/Downloadable-Public-Use-Files/SynPUFs/DE_Syn_PUF
4. Aditya Grover, Jure Leskovec, node2vec: Scalable Feature Learning for Networks